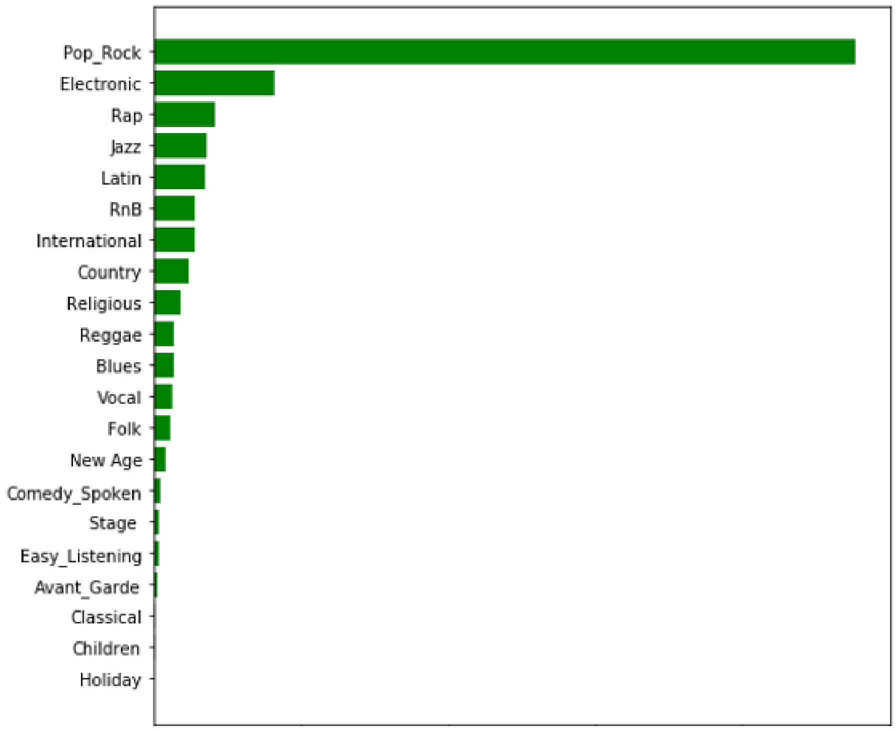
DATA420-24S1
Scalable Data

Assignment 2
The Million Song Dataset (MSD)



Student Name: Wijesingha Mayadunnage Uditha Iresh Mayadunna

Student ID: 39768450

1 Contents

1 Contents ... 2

1.1 Figures ... 3

1.2 Tables ... 3

2 Background ... 5

3 Data Processing ... 6

3.1 Question 1 ... 6

3.1.1 Data Structure ... 6

3.1.2 Number of rows in each dataset ... 7

3.2 Question 2 ... 7

3.2.1 Song-Track Mismatches ... 7

4 Audio similarity ... 8

4.1 Question 1 ... 8

4.1.1 Audio Feature Datasets ... 8

4.1.2 Data Set “msd-jmir-methods-of-moments-all-v1.0” ... 8

4.1.3 MSD All Music Genre Dataset (MAGD) ... 9

4.2 Question 2 ... 9

4.2.1 Binary Classification Algorithms ... 9

4.2.2 Transform the genre into a binary column ... 11

4.2.3 Training the Models ... 11

4.3 Question 3 ... 12

4.3.1 Hyperparameters ... 12

4.3.2 Cross validation ... 13

4.3.3 Hyperparameter Tuning ... 13

4.4 Question 4 ... 14

4.4.1 Logistic Regression for Multiclass Classification ... 14

5 Song Recommendations ... 16

5.1 Question 1 ... 16

5.1.1 Taste profile dataset ... 16

5.1.2 Distribution of Song Popularity and User Activity ... 16

5.2 Question 2 ... 16

5.2.1 Creating a clean data set ... 16

5.2.2 Splitting the data ... 17

5.2.3 Model training and user recommendations 17
5.2.4 Least Squares (ALS)Performance Metrics of the Model 19
5.2.5 Usefulness and Limitations of these metrics 19
5.2.6 Alternative Evaluation Methods 19
5.2.7 Additional Metrics for Future Plays 19
6 Conclusion 20
7 References 21
8 Appendix 22

8.1 Appendix 1 – Audio Feature Datasets 22

8.2 Appendix 2 – Descriptive Statistics of “msd-jmir-methods-of-moments-all-v1.0” data set 22

8.3 Appendix 3 – MAGD Genre Distribution 23

8.4 Appendix 4 – Users per Song and Songs per User 24

1.1 Figures

Figure 1: Data Directory of MSD Data Set 6

Figure 2: Correlation between feature columns 8

Figure 3: MASD Genre Distribution 9

Figure 5: Class imbalance and class wise performance in multiclass classification 14

Figure 6: Multiclass Performance after resampling 15

Figure 7: Total Plays per Song 16

Figure 8: Total Plays per User 16

Figure 9: Number of Users per Song 24

Figure 10: Number of Songs per User 24

1.2 Tables

Table 1 : MSD Sub Folder Size 7

Table 2: Number of rows and unique songs in each dataset 7

Table 3: Characteristics of Binary Classification Models 10

Table 4: Class balance for binary classification 11

Table 5: Train and Test Data 11

Table 6: Model Performance Metrics 12

Table 7: Default Hyperparameter Values 13

Table 8 : Model Performance for Multiclass Classification 14

Table 9: Model Performance for Multiclass Classification after resampling the training data 15

Table 10: User 1 Song Recommendations 18

Table 11: User 2 Song Recommendations 18

Table 12: ALS Model Evaluation 19

Table 13:Audio Feature Datasets 22

Table 14:Descriptive Statistics of “msd-jmir-methods-of-moments-all-v1.0” data set 23

Table 15:MAGD Genre Distribution 23

2 Background

The second assignment for the DATA420 Scalable Data course has entailed analysing a set of datasets known as the Million Song Dataset (MSD), which originated from a collaboration between The Echo Nest and LabROSA. The homepage of the Million Song Dataset, Welcome! | Million Song Dataset (Bertin-Mahieux, 2011) has offered comprehensive information about the datasets, covering important details like mismatched songs and other relevant facts. Although various datasets were available, this assignment has focused only on two main datasets: the Taste Profile and All Music datasets.

In this assignment, there were three main components. The first part involved exploring the datasets within the Million Song Dataset, examining their sizes, formats, data types, and how each dataset was stored in HDFS. This part also included calculating the number of rows in each dataset and determining all the unique songs in each dataset. These calculations provided a comprehensive understanding of the datasets which were useful in the other two components of the assignment.

The second part involved using one of the feature datasets to investigate audio similarity. This included employing numerical representations of an audio waveform from the audio features dataset to determine the genre of a song. Three types of binary classification models from the spark.ml library was selected and trained to predict the genre. Before training the models, the issue of class imbalance arose, and different resampling techniques were used to address it. In the binary classification section, only the "Electronic" genre was predicted, and the discussion covered the default hyperparameters of each model and the use of cross-validation to fine-tune these hyperparameters. Although hyperparameter tuning could improve model performance, it was not performed due to limited time and resources. After evaluating the binary classification models, a multiclass classification model was used to predict all the genres in the dataset. Resampling techniques were also applied here to address the issue of class imbalance.

The final part of the assignment focused on recommending songs using collaborative filtering with the Taste Profile dataset. Since the distribution plots of song popularity and user activity displayed heavy-tailed (Power Law) distributions, songs with fewer play counts and users with fewer play counts were removed from the original dataset. The Alternating Least Squares (ALS) method in the spark.ml library was then used to train an implicit matrix factorization model. Recommendations were generated using the test dataset, and three types of offline metrics were employed to evaluate the model's performance. This entire process provided significant insights and hands-on experience with machine learning algorithms in the Spark distributed computing framework.

All preprocessing, song genre prediction, and song recommendations were conducted using the Apache Spark distributed computing framework. This involved exploring the allocation of resources, creation of partitions, and task assignments for each stage. Additionally, Python coding, grammar corrections, and other information were supported by ChatGPT (OpenAI, 2024), Stack Overflow (Stack Exchange Inc;, 2024) and QuillBot (QuillBot, 2024).

3 Data Processing

3.1 Question 1

3.1.1 Data Structure

The Millions Song Dataset (MSD) is organized into four primary directories: audio, genre, main, and taste profile. Within the audio directory, there are three subfolders: attributes, features, and statistics, with the attributes and features subfolders each containing 13 CSV files. The genre directory includes three .tsv files, while the main directory features a summary subfolder which contains two.csv.gz files. The taste profile directory consists of a mismatch's subfolder and a triplets.tsv file. Figure 1 illustrates the directory structure of the Millions Song Dataset.

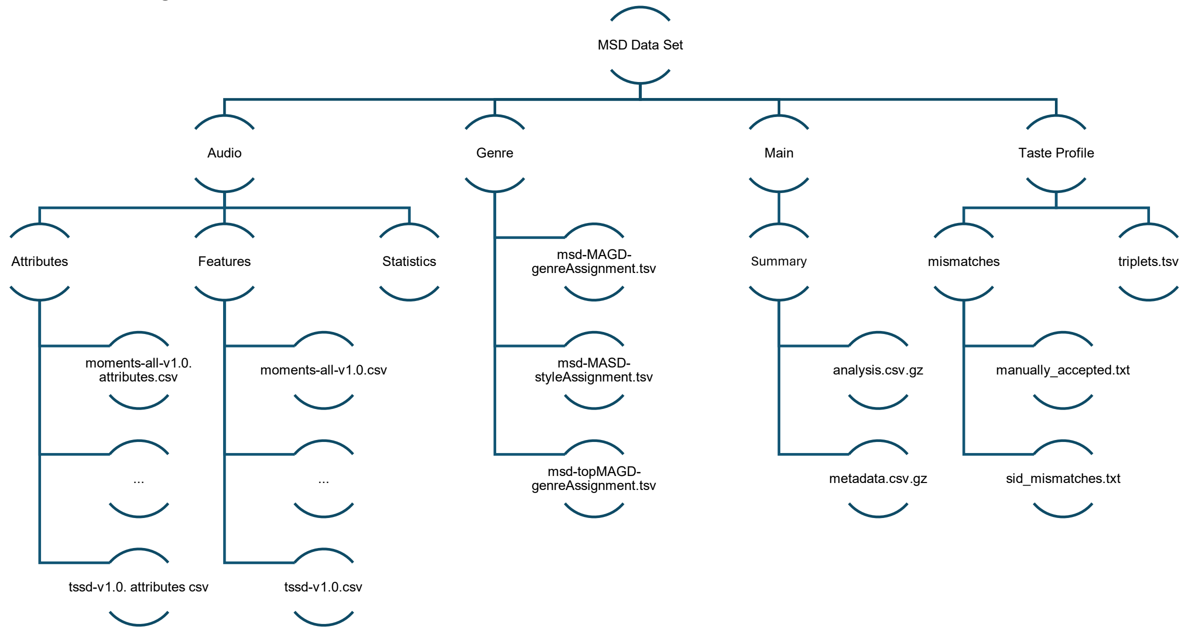


Figure 1: Data Directory of MSD Data Set

The MSD dataset originally occupies about 104 GB of space, but it is compressed to approximately 13 GB in the HDFS. Table 1 provides a comparison of the sizes of the original files and their compressed versions within the HDFS environment.

MSD data			
Sub Folder/File		Actual Size	Compressed Size
Audio	Attributes	824.3 KB	103.0 KB
	Features	97.8 GB	12.2 GB
	Statistics	322.1 MB	40.3 MB
Genre	-	241.0 MB	30.1 MB
Main	Summary	1.4 GB	174.4 MB
Taste Profile	Mismatches	16.2 MB	2.0 MB

	Triplets.tsv	3.8 GB	488.4 MB
Total		**103.53 GB**	**12.98 GB**

Table 1 : MSD Sub Folder Size

Table 1 indicates that the audio folder occupies the most significant amount of space within the dataset, with the features subfolder being the largest component.

3.1.2 Number of rows in each dataset

Table 2 shows the number of rows in each dataset and the number of unique songs featured in each dataset.

Sub Folder/File		**Number of Rows**	**Number of Unique Songs**
Audio	Attributes	3,929	-
	Features	12,927,867	994,176
	Statistics	992,865	992,865
Genre	Genre Assignment	422,714	422,714
	Style Assignment	273,936	273,936
	TopMAGD-genreAssignment	406,427	406,427
Main	Summary	2,000,000	998,964
Taste Profile	Mismatches	19,094	18,913
	Triplets.tsv	48,373,586	384,546

Table 2: Number of rows and unique songs in each dataset

According to Table 2, the triplets.tsv file contains the highest number of rows, totalling 48,373,586, but it includes only 384,546 unique songs. The summary folder details 998,964 unique songs, which is nearly 1 million. In the genre and statistics folders, the row count is equal to the number of unique songs.

3.2 Question 2

3.2.1 Song-Track Mismatches

According to the Million Song's blog (Bertin-Mahieux, 2011), some tracks in the Million Song Dataset were matched to the wrong songs in the Taste Profile dataset. In the second part of the assignment, audio similarity was examined using one of the feature datasets and all music genre datasets. In the third part of the assignment, songs were recommended using the Taste Profile dataset. Therefore, in the second part, only track IDs were dealt with, and in the third part, only song IDs were addressed. Consequently, no action was needed regarding the mismatched songs, as song IDs and track IDs did not overlap throughout the assignment.

4 Audio similarity

4.1 Question 1

4.1.1 Audio Feature Datasets

As outlined in section 4.1.1 of the processing details, the audio features subfolder includes 13 different datasets in .csv format. The specifics of these datasets are in appendix1. According to the table 9 in appendix 1, each dataset contains a similar number of rows, approximately around 994,000. However, the number of columns (features) varies significantly, ranging from a maximum of 1,140 to a minimum of 10. Consequently, the dataset with the fewest columns, "msd-jmir-methods-of-moments-all-v1.0" was chosen to determine audio similarity for this assignment.

4.1.2 Data Set “msd-jmir-methods-of-moments-all-v1.0”

The “msd-jmir-methods-of-moments-all-v1.0” has 994,623 rows and 10 feature columns and the descriptive statistics can be found in appendix 2.

4.1.2.1 Correlation between feature variables

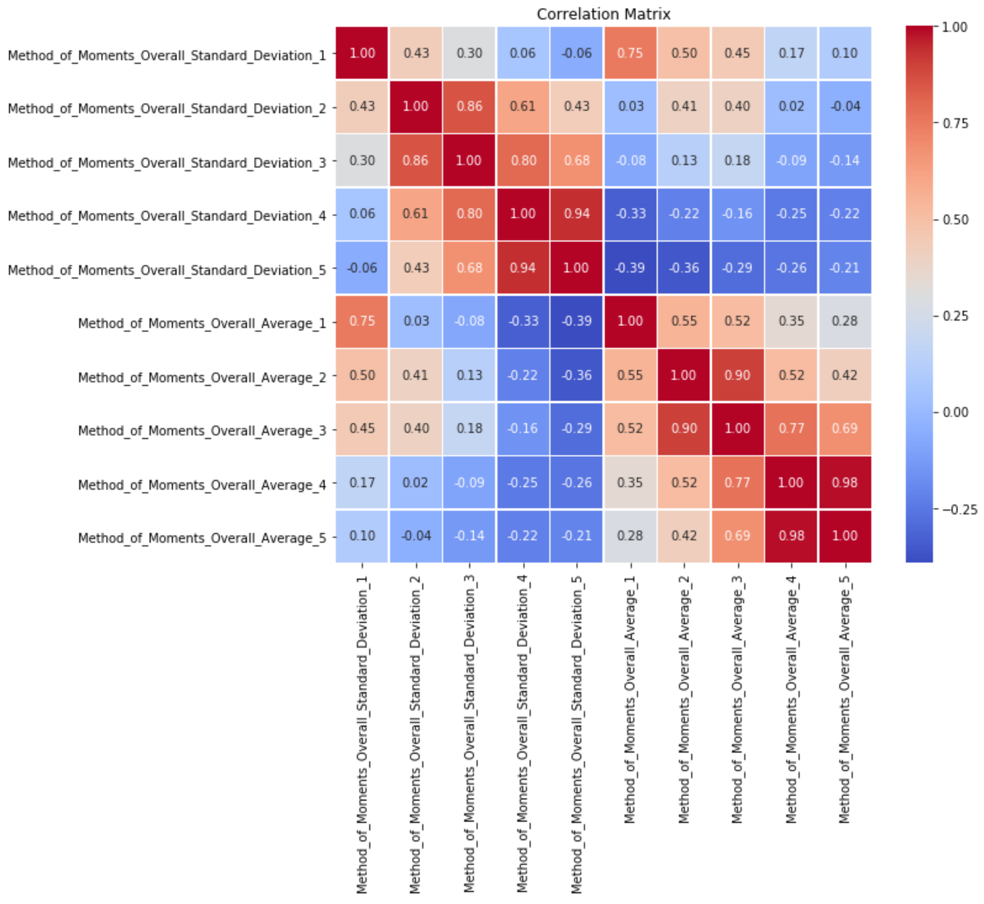


Figure 2: Correlation between feature columns

The correlation plot in Figure 2 illustrates the correlations between feature columns in the dataset. It reveals that the variables Method_of_Moments_Overall_Average_4 and Method_of_Moments_Overall_Average_5 are highly correlated, with a correlation value of 0.98. Similarly, Method_of_Moments_Standard_Deviation_4 and Method_of_Moments_Standard_Deviation_5 exhibit a high correlation value of 0.94. To address the issue of multicollinearity, a threshold of 0.95 was set, and one of the correlated feature columns was removed. Consequently, the Method_of_Moments_Overall_Average_5 feature column was dropped from the dataset before training the model.

4.1.3 MSD All Music Genre Dataset (MAGD)

The MSD All Music Genre Dataset (MAGD) comprises 422,714 rows, categorized into 21 different genres. The figure 2 illustrate the distribution of genres for the matched songs.

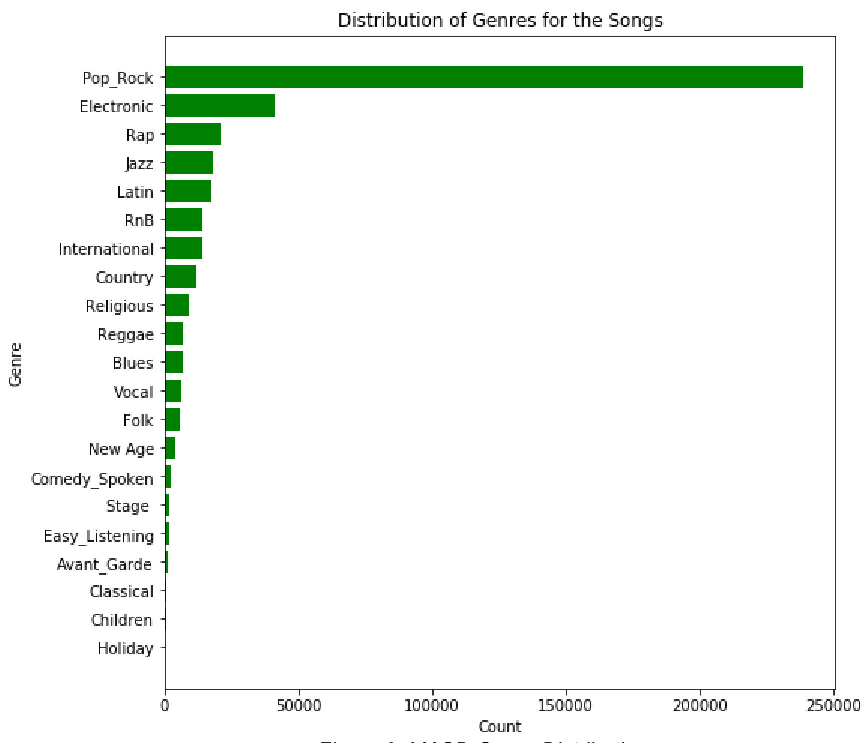


Figure 3: MASD Genre Distribution

Figure 3 shows that the songs are not evenly distributed across all genres. There are more than 200,000 pop/rock songs, making it the most represented genre, followed by around 50,000 electronic songs. Some genres have fewer than 1,000 songs, with the exact numbers detailed in Appendix 3.

4.2 Question 2

4.2.1 Binary Classification Algorithms

When choosing binary classification algorithms from the spark.ml library, three different types of models were selected to represent distinct methodologies. Logistic Regression was chosen as a Generalized Linear Model (GLM) method, the Gradient Boosted Tree Classifier was selected as an ensemble method, and the Support Vector Machine (SVM) was chosen as a kernel method. The following table shows the characteristics of each of the selected models.

Model	Model Flavour	Globally Transparent	Fussy	Eager	Parametric	Fast to train	Non linear
Logistic Regression	Generalized Linear Model	Yes	Yes	Yes	Yes	Yes	No
Gradient Boosted Tree	Ensemble method	No	No	No	No	No	Yes
Support Vector Machine	Kernel method	No	No	Yes	No	No	Yes

Table 3: Characteristics of Binary Classification Models

4.2.1.1 Logistic Regression

Logistic regression is an efficient and easily interpretable model for binary classification. Its coefficients provide clear insights into the relationships between predictors and the outcome. It performs well with linear relationships and is computationally efficient, making it suitable for large datasets. The model requires minimal hyperparameter tuning, mainly focusing on regularization to address multicollinearity and high-dimensional data. Although it may not offer the highest predictive accuracy for complex, non-linear relationships, its simplicity, clarity, and ease of use make it a strong choice, particularly when interpretability and quick training are crucial (GeeksforGeeks, 2024).

4.2.1.2 Gradient Boosted Tree

Gradient Boosted Trees (GBT) are powerful and flexible models known for their high predictive accuracy, particularly in complex, non-linear datasets. GBT is not a globally transparent model, and therefore cannot be interpreted in the same way as transparent models such as logistic regression. Training GBTs can be computationally expensive and slower due to their iterative nature, but the resulting model often justifies the cost with superior performance. GBT hyperparameters include the number of trees, learning rate, and tree depth, which can be costly to tune but are required for best performance. GBTs excels at handling high-dimensional data and automatically capturing feature interactions. However, they may encounter scalability issues with excessively large datasets, which can be mitigated using distributed computing frameworks such as Apache Spark (Ibrahim, n.d.).

4.2.1.3 Support Vector Machine

Support Vector Machines (SVMs) are strong classifiers noted for their excellent predicted accuracy, particularly in high-dimensional spaces, as well as their ability to handle non-linear interactions with kernel functions. However, as compared to simpler models such as logistic regression, SVMs might be more difficult to analyse and understand. Training SVMs may be computationally costly and time consuming, especially with huge datasets, yet they frequently produce a very accurate decision boundary. Optimal performance in SVMs necessitates careful hyperparameter tweaking, such as choosing the appropriate kernel and regularisation parameter, which may be challenging and time-consuming. SVMs excel at managing high-dimensional data and effectively prevent overfitting by maximising the margin between classes. However, they may struggle with very large datasets, where scaling and memory usage become significant concerns. Techniques like using linear SVMs or stochastic gradient descent can help mitigate these issues (GeeksforGeeks, 2023).

4.2.2 Transform the genre into a binary column

After selecting three binary classification models, the next step was to transform the genre column into a binary format. Before that, as mentioned earlier, one of the highly correlated columns “Method_of_Moments_Overall_Average_5” was removed from the dataset. Then, the genre "Electronic" was labelled as 1, and all other genres were labelled as 0. The class balance of the binary labels is as follows.

GENRE	LABEL	COUNT	RATIO
ELECTRONIC	1	40,666	0.096681
OTHER	0	379,954	0.903319
TOTAL		420,620	

Table 4: Class balance for binary classification

4.2.3 Training the Models

4.2.3.1 Splitting the data

Before the model training phase began, the dataset was partitioned into training and testing sets using a stratified random sampling technique, ensuring that each subset maintained the same proportion of classes as the original dataset. The training set comprised 80% of the data, while the testing set contained the remaining 20%. The following table shows the training and testing data count after splitting the data.

		TOTAL (100%)		**TRAINING DATA (80%)**		**TEST DATA (20%)**	
GENRE	Label	Count	Ratio	Count	Ratio	Count	Ratio
ELECTRONIC	1	40,666	0.096681	32,532	0.096679	8,134	0.096689
OTHER	0	379,954	0.903319	303,963	0.903321	75,991	0.903311
TOTAL		420,620		336,495		84,125	

Table 5: Train and Test Data

4.2.3.2 Resampling Methods

To handle the class imbalance in the binary classification problem, different strategies like downsampling, upsampling, and observation weighting were implemented on the training set. Subsequently, all three models were trained using the resampled data to assess their performance across various imbalance correction techniques.

4.2.3.3 Model Training and Performance

The three models were trained using different resampling methods, and their performance for each technique is summarized as follows.

Model	Resampling Technique	Precision	Recall	Accuracy	Auroc
Logistic Regression	No Sampling	0.4754	0.0226	0.9030	0.7548
	Down Sampling	0.2933	0.4192	0.8462	0.7552
	Up Sampling	0.2977	0.4151	0.8487	0.7553
	Observation reweighting	0.1854	0.7215	0.6667	0.7549
Gradient Boosted Tree	No Sampling	0.6189	0.0946	0.9068	0.7994
	Down Sampling	0.3310	0.4879	0.8551	0.8023
	Up Sampling	0.3320	0.4893	0.8554	0.8040
	Observation reweighting	0.2251	0.7231	0.7326	0.8041
Support Vector Machine	No Sampling	0.0000	0.0000	0.9033	0.7067
	Down Sampling	0.4440	0.0346	0.9024	0.7178
	Up Sampling	0.4407	0.0759	0.9013	0.7139
	Observation reweighting	0.1739	0.7023	0.6486	0.7252

Table 6: Model Performance Metrics

As indicated in Table 9, the gradient boosted tree model achieved the highest AUROC across all resampled training instances. When resampling techniques were applied to the gradient boosted tree method, there was a slight increase in the AUROC value. Additionally, precision and recall became more balanced, though there was a significant drop in accuracy.

In all three models, the highest accuracy was observed when no resampling techniques were applied, though the precision and recall were not balanced. In the case of the support vector machine, without any resampling techniques, both true positive and false positive values were 0, resulting in precision and recall values of zero. Despite this, the accuracy was 0.9033, highlighting that accuracy is not a reliable metric for model comparison in the presence of significant class imbalance.

4.3 Question 3

4.3.1 Hyperparameters

As previously mentioned, the models employed have unique characteristics and different hyperparameters. The table below outlines few hyperparameters for each model along with their default values.

Model	Hyperparameter	Default Value
Logistic Regression	maxIter	100
	regParam (Lambda)	0.0
	elasticNetParam	0.0
	threshold	0.5
	standardization	True
Gradient Boosted Tree	maxIter	20

	maxDepth	5
	stepSize	0.1
	minInstancesPerNode	1
	minInfoGain	0.0
Support Vector Machine	maxIter	100
	regParam (Lambda)	0.0
	tol	1e-6
	fitIntercept	True
	standardization	True

Table 7: Default Hyperparameter Values

For logistic regression, the regParam would be tuned in order to introduce regularization, which would prevent overfitting and potentially improve generalization. Additionally, the maxIter parameter can be adjusted if the model did not converge within the default 100 iterations and that could improve the performance of the model (Gusarov, 2022).

For GBT, maxIter can be increased beyond 20 which might improve performance, especially for more complex datasets. But it would increase the risk of overfitting and computation time. It is crucial to modify maxDepth in order to find a balance between preventing overfitting and capturing complex patterns. If the training is unstable, the stepSize can be decreased to gradually fine-tune the model (Jain, 2022).

It is important to experiment with different regParam (C) values for LinearSVC in order to find a balance between margin the optimisation and classification error. If the model fails to converge in the allocated 100 iterations, then maxIter may need to be increased (Fraj, 2018).

4.3.2 Cross validation

Cross-validation is a technique for evaluating a model's performance and generalizability that involves splitting the dataset into k subsets (folds). The model is trained on k-1 folds and validated on the remaining folds. This method is repeated k times, with each fold serving as the validation set once. The performance metrics from each iteration are then averaged to obtain the final performance metrics of the model.

This method is especially valuable for hyperparameter tuning because it ensures that the chosen hyperparameters perform well across different subsets of data. This reduces the risk of overfitting and results in a model that generalizes better to unseen data. By using cross-validation, we make efficient use of the entire dataset and this approach overall, provides a more reliable assessment of the model's performance (Brownlee, 2023).

4.3.3 Hyperparameter Tuning

The hyperparameter tuning process begins with specifying an objective measure, such as accuracy, F1-score or another metric. Next, a hyperparameter grid is created to explore using methods like grid search, random search, or Bayesian optimization. Using k-fold cross-validation, the model's performance is evaluated for each hyperparameter combination, ensuring robust generalization. The best-performing set of hyperparameters is selected based on average performance metrics across the folds. Model performance can be significantly improved by hyperparameter tuning and Metrics can be improved by many percentage points, particularly if the original model setup was not optimal (Pandian, 2024).

4.4 Question 4

4.4.1 Logistic Regression for Multiclass Classification

Logistic regression can be extended to multiclass classification using the One-vs-Rest (OvR) approach. This method involves training a separate binary classifier for each class to distinguish it from the others. For a dataset with k classes, k binary logistic regression models are trained, and each model classifies instances of its specific class against all other classes. During prediction, each of the k classifiers produces a probability score indicating the likelihood that a new instance belongs to their respective class. The class with the highest probability score among all classifiers is selected as the final predicted class. This method effectively extends logistic regression to handle multiclass classification tasks by leveraging multiple binary classifiers (Pramoditha, 2023).

4.4.1.1 Model Training and Performance

The model was trained to capture each class and the overall performance is as follows.

Model	Accuracy	Precision	Recall	F1 Score	Test Error
Logistic Regression	0.5711	0.5808	0.9799	0.4392	0.4288

Table 8 : Model Performance for Multiclass Classification

When comparing with binary classification, there is a significant decline in the accuracy metric for multiclass classification. Additionally, upon extracting class-wise metrics, it becomes evident that only the classes with more training data are well-trained, while classes with fewer observations exhibit zero precision and recall values. This indicates that the model struggles to learn effectively for underrepresented classes, leading to poor performance in predicting these minority classes. The following figure shows how class imbalance affects this multiclass model performance.

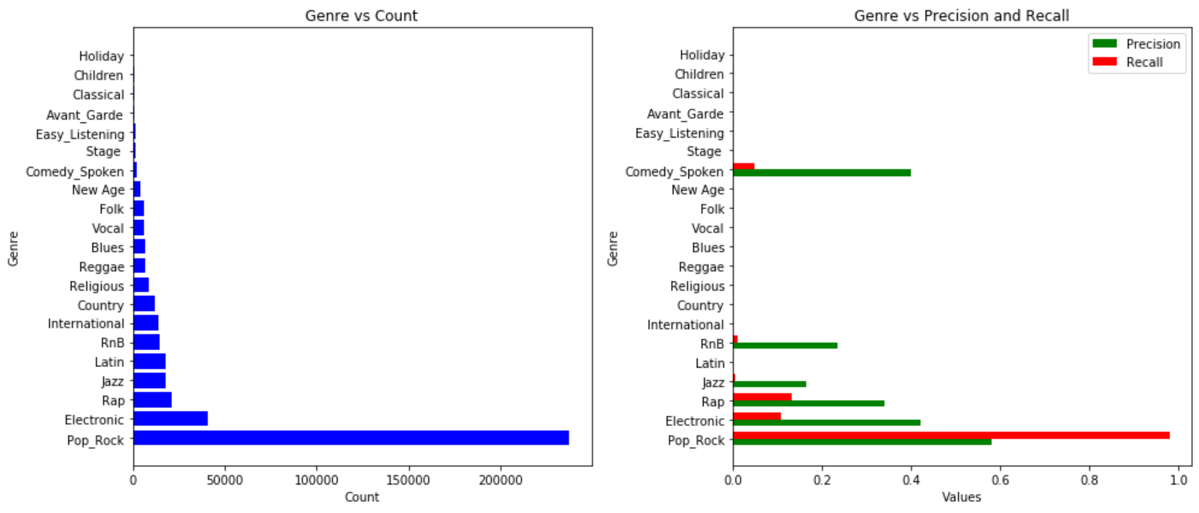


Figure 4: Class imbalance and class wise performance in multiclass classification

4.4.1.2 Resampling Observations

To address the class imbalance issue, the training observations of minority classes were up sampled while classes with more observations were down sampled. The following table illustrates the model's performance after applying these resampling techniques.

Model	Accuracy	Precision	Recall	F1 Score	Test Error
Logistic Regression	0.5421	0.6432	0.854849	0.4686	0.4579

Table 9: Model Performance for Multiclass Classification after resampling the training data

According to the table, it is evident that although the overall accuracy has decreased, the F1 score has increased after applying resampling techniques. The most important factor is the significant improvement in class-wise metrics such as precision and recall for most classes. The figure 4 illustrates the changes in class-wise precision and recall after resampling the training data, highlighting the enhanced performance in previously underrepresented classes.

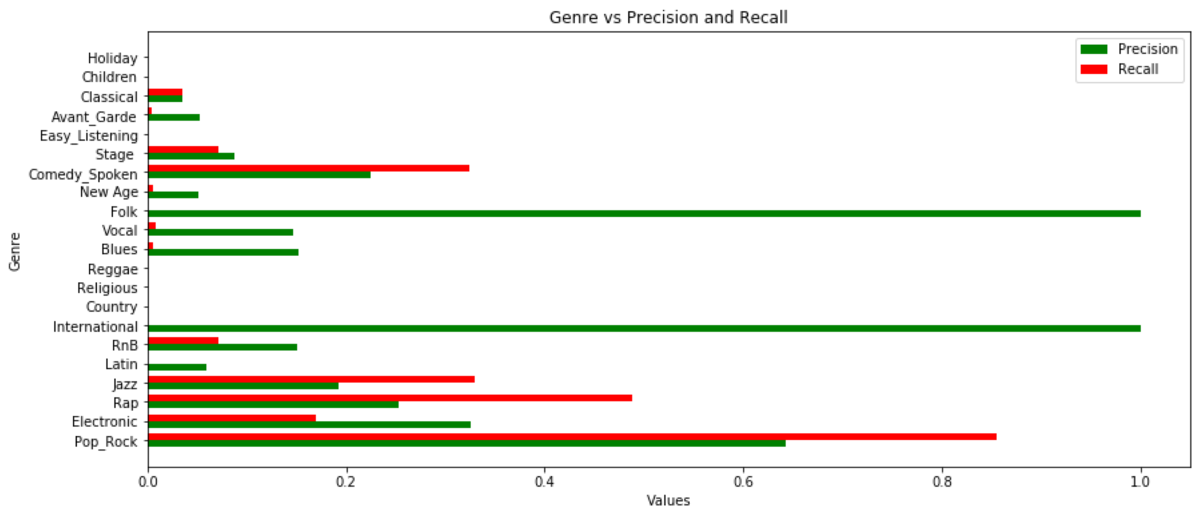


Figure 5: Multiclass Performance after resampling

5 Song Recommendations

5.1 Question 1

5.1.1 Taste profile dataset

The taste profile dataset, which is 488.4 MB in compressed form, contains 48,373,586 rows. This dataset is stored as eight csv.gz files in HDFS. Since the data set is not that huge, cashing and repartitioning would improve the efficiency when training a collaborative filtering model.

The triplet's dataset contains data on 384,546 unique songs and 1,019,318 unique users. The most active user, identified as "093cb74eb3c517c5179ae24caf0ebec51b24d2a2," has played 202 different songs a total of 13,132 times. This user's played songs represent approximately 0.053% of the total number of unique songs in the dataset.

5.1.2 Distribution of Song Popularity and User Activity

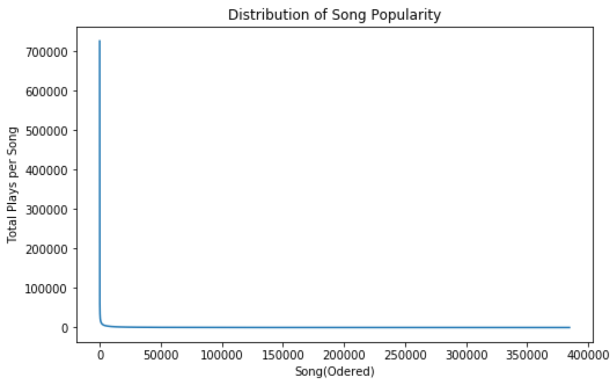


Figure 6: Total Plays per Song

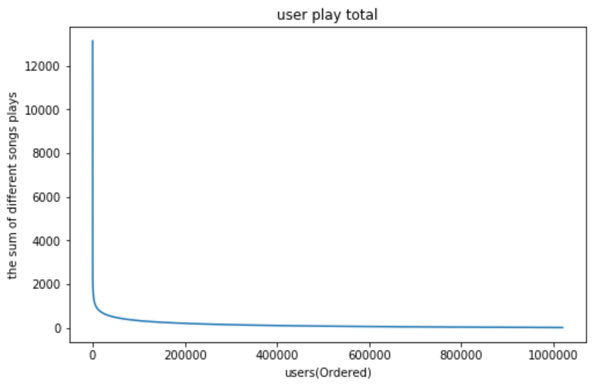


Figure 7: Total Plays per User

Figures 4 and 5 illustrate the distribution plots for songs and users, respectively. Both figures show a heavy-tailed distribution (power law distribution), indicating that the majority of songs have been played only a few times and most users have listened to only a few songs. Additionally, Appendix 6 provides further evidence by illustrating the number of users per song and the number of songs per user, both of which depict the same type of distribution.

5.2 Question 2

5.2.1 Creating a clean data set

Given the heavy-tailed distribution highlighted in previous graphs, many songs are seldom played, and many users have limited interactions with different songs. If we split the data into training and testing sets in an 80/20 ratio, it's likely that over half of the users in the test set will have listened to only a few songs. This can significantly limit the effectiveness of the model because these users will have minimal historical data to inform predictions. To address this issue, mean value of total plays per song was calculated and filtered all the songs lower to the mean value play count of 360. The same thing did with the total plays per user and users lower to the mean value play count of 136 was removed. So, this model will not be suitable for collaborative recommendations for the infrequent users and those users can be recommended with popularity recommendations.

5.2.2 Splitting the data

The dataset was divided into a 75/25 train-test split, with the test set containing 21,691,543 play counts, which is 25.03% of the total play counts. It's crucial for every user in the test set to have some interactions in the training set as well, as this ensures that recommendations can be generated, and the model's performance can be accurately assessed. There are two potential solutions to address this issue. The first is to overlook the problem, which would result in null recommendations for users who are not present in both the training set. The second solution is to perform an anti-join between the test and train datasets to exclude users from the test set who do not appear in the training set. In this particular assignment, the solution one is applied.

5.2.3 Model training and user recommendations

After an implicit matrix factorization model was trained using Alternating Least Squares method, two users from the test set were chosen, and recommendations were generated for them. The following tables provide the relevant songs for each user and the recommendations made by the model to each user.

5.2.3.1 User 1

User Code = 28

Relevant Songs			Recommended Songs		
Song Title	**Artist Name**	**Genre**	**Song Title**	**Artist Name**	**Genre**
Hollaback Girl	Gwen Stefani	Pop Rock	**All The Right Moves**	OneRepublic	Pop Rock
My Happy Ending	Avril Lavigne	Pop /Rock	**Lucky**	Jason Mraz	Folk-Pop
Rock The House	Gorillaz	Alternative Hip-Hop	**Billionaire**	Travie McCoy	Pop Rap
Sin Despertar	Kudai	Latin Pop	**Bring Me to Life**	Evanescence	Alternative Metal
Dare	Gorillaz	Electronic	**The Scientist**	Coldplay	
Teddy Picker	Arctic Monkeys	Indie Rock	**Kryptonite**	3 Doors Down	Alternative Rock
The Only Differen	Panic! At The Disco	Pop Punk	**Use Somebody**	Kings Of Leon	Alternative Rock
Charlotte	Air Traffic	Indie Rock	**Secrets**	OneRepublic	Pop Rock
When The Day Met	Panic! At The Disco	Pop Rock	**Sehr kosmisch**	Harmonia	Electronic
Solo un segundo	Bacilos	Pop Rock	**Fireflies**	Charttraxx	Electropop

				Karaoke	
This House is a	Arctic Monkeys	Indie Rock	**The Only Exception**	Paramore	Alternative Rock

Table 10: User 1 Song Recommendations

According to Table 9, the majority of the songs in the relevant list for user 1 are in the Rock genre, which includes subgenres such as Pop Rock and Indie Rock (a subgenre of Alternative Rock). Additionally, five of the recommendations are in the same genre, suggesting that the model's recommendations identify the type of music that the particular user prefers.

5.2.3.2 User 2

User Code = 600

Relevant Songs			Recommended Songs		
Song Title	**Artist Name**	**Genre**	**Song Title**	**Artist Name**	**Genre**
All The Right Moves	OneRepublic	Pop Rock	**Halo**	Beyoncé	Pop, R&B
Livin' On A Prayer	Bon Jovi	Pop Rock	**Dog Days Are Over**	Florence	Indie Rock
One Less Lonely Girl	Justin Bieber	Pop, R&B	**Billionaire**	Travie McCoy	Pop Rap
We Don't Stop	Young Bleed	Hip-Hop	**Pursuit Of Happiness**	Kid Cudi	Hip-Hop
Ironmasters	The Men They Coul	Folk Rock	**Secrets**	OneRepublic	Pop Rock
Brave The Elements	Colossal	Indie Rock	**Le Courage Des**	Dominique A	Indie Pop
Seppuku	Beep Beep	Indie Rock	**Sehr kosmisch**	Harmonia	Electronic
Fireflies	Charttraxx Karaoke	Electropop	**Fireflies**	Charttraxx Karaoke	Electropop
You Belong with Me	Taylor Swift	Country/Pop	**The Only Exceptio**	Paramore	Alternative Rock
Love Story	Taylor Swift	Country/Pop	**Bulletproof**	La Roux	Electropop
Heaven's Missing	98	Pop, R&B	**Love Story**	Taylor Swift	Country/Pop

Table 11: User 2 Song Recommendations

According to Table 10, two songs from the user 2's relevant list also appear in the recommended list: "Fireflies" by Charttraxx Karaoke and "Love Story" by Taylor Swift. Both the relevant and recommended song genres are somewhat similar, indicating that the model's recommendations align with the user's music preferences.

By analysing the recommendations for the two users, it was found that there were similarities between the relevant and recommended lists. This indicated that the model effectively identified and matched the users' preferred music to a satisfactory level.

5.2.4 Least Squares (ALS)Performance Metrics of the Model

After training the model, test data was used to make recommendations and the model was evaluated using three different offline methods, Precision @ 10, NDCG @ 10 and Mean Average Precision (MAP) @10. The metrics values are as follow.

Metric	**Value**
Precision @ 10	0.04705
NDCG @ 10	0.05353
Mean Average Precision (MAP) @10	0.02112

Table 12: ALS Model Evaluation

5.2.5 Usefulness and Limitations of these metrics

The main advantage of offline evaluation methods is their ability to easily scale to large datasets. This scalability makes them particularly useful during the initial phases of model development, where a vast amount of historical data is available, and the data is less dynamic. In this assignment, there was a large dataset, and the model was easily evaluated using these offline metrics.

However, offline methods have several limitations: First, the data is static, meaning the fixed dataset used for evaluation may not accurately reflect evolving user preferences or behaviours. Second, there is a lack of real-time user interaction, so offline methods cannot capture the dynamic nature of user engagement, potentially resulting in suboptimal performance. Third, offline methods may introduce bias, as they rely on historical user experience data that may not represent current or future user dynamics, ignoring the fact that people's preferences can change over time.

5.2.6 Alternative Evaluation Methods

A/B Testing is an effective alternative method, comparing two recommendation services by directly measuring user engagement (clicks, plays) and satisfaction (ratings, feedback) in a real-world setting. This approach offers practical insights into how different recommendation strategies affect actual user behaviour, providing a more comprehensive evaluation of their effectiveness. The A/B test is an online method of evaluating recommended models and it should be trained regularly with the new data.

5.2.7 Additional Metrics for Future Plays

When evaluating future user-song plays based on recommendations, it is important to consider user satisfaction, user engagement, diversity of recommendations, and novelty of recommendations. Given that many song recommender systems tend to be biased towards past user behaviour, examining diversity and novelty can provide insights into this bias.

6 Conclusion

This assignment focused on finding audio similarity using one of the feature datasets from the Million Song Dataset (MSD) and providing recommendations using the triplet's dataset from MSD. Initially, the data was processed, revealing that the MSD contains various subfolders and file types, making it crucial to understand the entire dataset before conducting any analysis. Although the total number of songs is roughly 1 million unique tracks, not all datasets within MSD include all these songs. For instance, the triplet's dataset contains only 384,546 unique songs. Despite some mismatches between song IDs and track IDs, this did not pose an issue for the assignment since the song IDs and track IDs did not overlap.

In the audio similarity part of the assignment, the genre of songs was predicted using their features, and three binary classification models were trained to distinguish between "Electronic" and other genres. The models used were Logistic Regression, Gradient Boosted Trees, and SVM. Gradient Boosted Trees performed the best on the AUROC metric with default hyperparameter values. Due to significant class imbalance, three resampling techniques were applied, resulting in a slight performance improvement. Although hyperparameter fine-tuning could further enhance model performance, it was not pursued due to time and resource constraints, and only the hyperparameters were discussed. Following the binary classification, a multiclass classifier was trained using a Logistic Regression classifier, which performed poorly compared to the binary classification. The class imbalance resulted in inadequate training for some classes, leading to poor performance on unseen data. While resampling techniques were applied to address class imbalance, they did not significantly improve overall performance.

The song recommendation model was created using the triplet's dataset. Analysis revealed that the majority of users are infrequent listeners, and most songs are played only a few times. Therefore, user and song filtration was performed before training the Alternating Least Squares model. The model's performance was evaluated using three different offline evaluation methods. Additionally, a few users were selected to compare relevant and recommended songs. For these users, the model recommended songs that closely matched their taste, indicating that the model works at a satisfactory level.

This assignment thoroughly explored various aspects of machine learning and how it can be implemented in a Spark distributed environment. It also confirmed the "No Free Lunch" theorem in machine learning, highlighting that no single model works best for every problem. There are many aspects to fine-tune to build a solid machine learning model, emphasizing the importance of carefully adjusting and optimizing different parameters and techniques to achieve the best performance.

7 References

Apache Spark. (n.d.). *JOIN*. Retrieved from Apache Spark - A Unified engine for large-scale data analytics: https://spark.apache.org/docs/latest/sql-ref-syntax-qry-select-join.html

Bertin-Mahieux, T. (2011). *Million Song Dataset*. Retrieved from Million Song Dataset: http://millionsongdataset.com/

Brownlee, J. (2023, October 04). *A Gentle Introduction to k-fold Cross-Validation*. Retrieved from machinelearningmastery: https://machinelearningmastery.com/k-fold-cross-validation/

Fraj, M. B. (2018, January 6). *In Depth: Parameter tuning for SVC*. Retrieved from Medium: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769

GeeksforGeeks. (2023, June 10). *Support Vector Machine (SVM) Algorithm*. Retrieved from Support Vector Machine (SVM) Algorithm: https://www.geeksforgeeks.org/support-vector-machine-algorithm/

GeeksforGeeks. (2024, April 11). *Logistic Regression in Machine Learning*. Retrieved from Logistic Regression in Machine Learning: https://www.geeksforgeeks.org/understanding-logistic-regression/

Gusarov, M. (2022, April 10). *Do I need to tune logistic regression hyperparameters?* Retrieved from Medium: https://medium.com/codex/do-i-need-to-tune-logistic-regression-hyperparameters-1cb2b81fca69

Ibrahim, M. (n.d.). *An Introduction to Gradient Boosted Trees for Machine Learning*. Retrieved from https://wandb.ai/mostafaibrahim17/ml-articles/reports/An-Introduction-to-Gradient-Boosted-Trees-for-Machine-Learning--Vmlldzo2NTQ4NzYx#:~:text=Gradient%2Dboosted%20trees%20(GBT),built%20one%20at%20a%20time.

Jain, A. (2022, June 15). *Complete Machine Learning Guide to Parameter Tuning in Gradient Boosting (GBM) in Python*. Retrieved from https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/

OpenAI. (2024). *ChatGPT (4.0) [Software]*. Retrieved from https://www.openai.com/

Pandian, S. (2024, May 25). *A Comprehensive Guide on Hyperparameter Tuning and its Techniques*. Retrieved from https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/

Pramoditha, R. (2023, November 20). *Logistic Regression for Multiclass Classification — 3 Strategies You Need to Know*. Retrieved from Medium: https://rukshanpramoditha.medium.com/logistic-regression-for-multiclass-classification-3-strategies-you-need-to-know-0a3e74574b96

QuillBot. (2024). *QuillBot Paraphrasing Tool [Software]*. Retrieved from https://www.quillbot.com/

8 Appendix

8.1 Appendix 1 – Audio Feature Datasets

FILE NAME	NUMBER OF ROWS	NUMBER OF FEATURE COLUMNS
MSD-JMIR-AREA-OF-MOMENTS-ALL-V1.0	994,623	20
MSD-JMIR-LPC-ALL-V1.0	994,623	20
MSD-JMIR-METHODS-OF-MOMENTS-ALL-V1.0	994,623	10
MSD-JMIR-MFCC-ALL-V1.0	994,623	26
MSD-JMIR-SPECTRAL-ALL-ALL-V1.0	994,623	16
MSD-JMIR-SPECTRAL-DERIVATIVES-ALL-ALL-V1.0	994,623	16
MSD-MARSYAS-TIMBRAL-V1.0	995,001	124
MSD-MVD-V1.0	994,188	420
MSD-RH-V1.0	994,188	60
MSD-RP-V1.0	994,188	1440
MSD-SSD-V1.0	994,188	168
MSD-TRH-V1.0	994,188	420
MSD-TSSD-V1.0	994,188	1,176

Table 13:Audio Feature Datasets

8.2 Appendix 2 – Descriptive Statistics of "msd-jmir-methods-of-moments-all-v1.0" data set

Feature	Count	Mean	Stander Deviation	Min	Max
Method_of_Moments_Overall_Standard_Deviation_1	994,623	0.15	0.066	0.0	0.959
Method_of_Moments_Overall_Standard_Deviation_2	994,623	10.38	3.86	0.0	55.42
Method_of_Moments_Overall_Standard_Deviation_3	994,623	526.8	180.43	0.0	2919.0
Method_of_Moments_Overall_Standard_Deviation_4	994,623	35071.9	12806.81	0.0	407100.0
Method_of_Moments_Overall_Standard_Deviation_5	994,623	5297870.36	2089356.43	0.0	4.657E7

Method_of_Moments_Overall_Average_1	994,623	0.35	0.18	0.0	2.647
Method_of_Moments_Overall_Average_2	994,623	27.46	8.35	0.0	117.0
Method_of_Moments_Overall_Average_3	994,623	1495.80	505.89	0.0	5834.0
Method_of_Moments_Overall_Average_4	994,623	143165.46	50494.27	-146300.0	452500.0
Method_of_Moments_Overall_Average_5	994,623	2.39 E7	9307340.2	0.0	9.477 E7

Table 14:Descriptive Statistics of "msd-jmir-methods-of-moments-all-v1.0" data set

8.3 Appendix 3 – MAGD Genre Distribution

Genre	**Count**
Pop_Rock	238,786
Electronic	41,075
Rap	20,939
Jazz	17,836
Latin	17,590
RnB	14,335
International	14,242
Country	11,772
Religious	8,814
Reggae	6,946
Blues	6,836
Vocal	6,195
Folk	5,865
New Age	4,010
Comedy_Spoken	2,067
Stage	1,614
Easy_Listening	1,545
Avant_Garde	1,014
Classical	556
Children	477
Holiday	200

Table 15:MAGD Genre Distribution

8.4 Appendix 4 – Users per Song and Songs per User

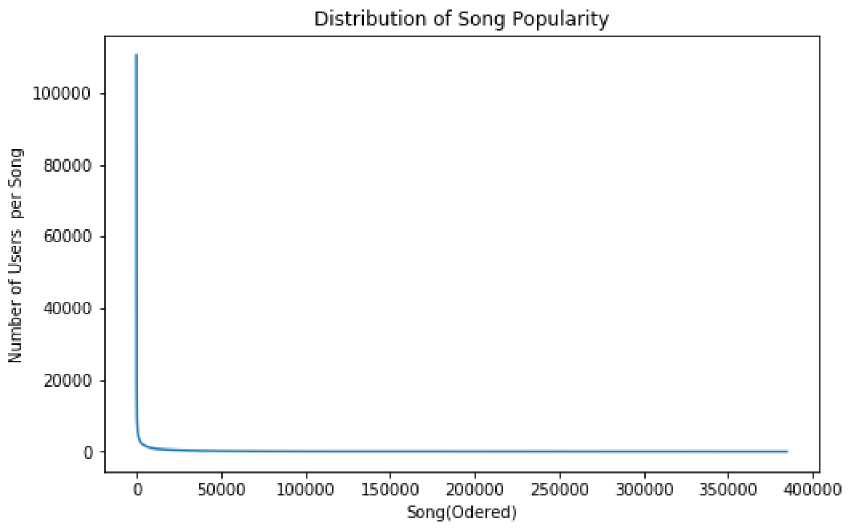


Figure 8: Number of Users per Song

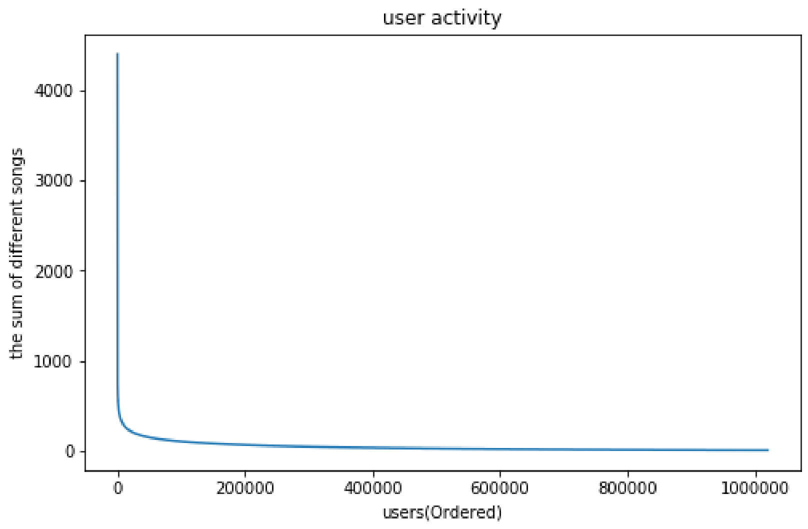


Figure 9: Number of Songs per User